6.előadás: Adatbázisok-I.

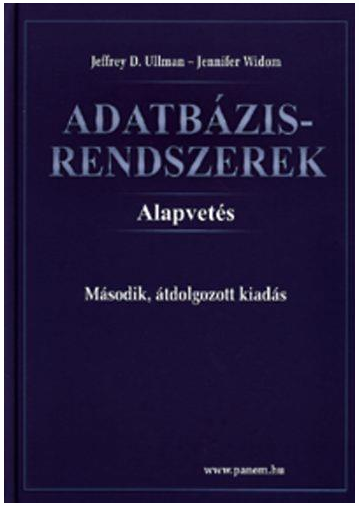
dr. Hajas Csilla (ELTE IK)
http://sila.hajas.elte.hu/

SQL gyakorlatban: SQL DML, SQL DDL

6.5. Változtatások az adatbázisban:
SQL DML adatkezelő utasítások:
INSERT, DELETE, UPDATE

[6.6. Kieg.: Tranzakciók, COMMIT]

7.fej. Táblák és megszorítások létrehozása
CREATE TABLE, CONSTRAINTS
(Folyt.köv.7ea: triggerek, nézettáblák)



SQL története, szabványok

- Szabvány adatbázis-kezelő nyelv: SQL
- SQL (angol kiejtésben SEQUEL) uis az SQL előfutára
  IBM fejlesztette ki a 70-es években: SEQUEL → SQL
  más is volt pl. Ingres : QUEL (ez kalkulus alapú lekérdezés)
- Szabványok (ANSI, ISO)
  SQL86, SQL89, SQL92 (SQL2), SQL:1999 (SQL3),
  SQL: 2003, SQL:2006, SQL:2008
- Nyelvjárások (Oracle, Sybase, DB2, Progress, MSSQL, mySQL, SQL Server, PostgreSQL, Access,…)
- Az SQL megvalósítások között vannak különbségek, gyakorlatokon az Oracle SQL-t nézzük meg részletesen.

SQL fő komponensei

- Az SQL elsődlegesen lekérdező nyelv (Query Language)
  SELECT utasítás (az adatbázisból információhoz jussunk)
- Adatkezelő nyelv, DML (Data Manipulation Language)
  INSERT, UPDATE, DELETE, SELECT
- Sémaleíró nyelv, DDL (Data Definition Language)
  CREATE, ALTER, DROP
- Adatvezérlő nyelv, DCL (Data Control Language)
  GRANT, REVOKE
- Tranzakció-kezelés
  COMMIT, ROLLBACK, SAVEPOINT
- Procedurális kiterjesztések
  SQL/PSM és a gyakorlatban Oracle PL/SQL

SQL DML
Adatbázis tartalmának módosítása

Tankönyv 6.5. Változtatások az adatbázisban

- A módosító utasítások nem adnak vissza eredményt, mint a lekérdezések, hanem az adatbázis tartalmát változtatják meg.
- 3-féle módosító utasítás létezik:

  INSERT - sorok beszúrása

  DELETE – sorok törlése

  UPDATE – sorok komponensei értékeinek módosítása

Beszúrás (insert into)

- Két alakja van:
- 1.) ha egyetlen sort szúrunk be:

  INSERT INTO <reláció>

  VALUES (<konkrét értékek listája>);

- 2.) ha több sort, egy lekérdezés eredményét visszük fel alkérdés segítségével:

  INSERT INTO <reláció>

  (<alkérdés>);

Beszúrás, attribútumok megadása

- Példa: A Szeret táblába beírjuk, Zsu szereti a Bud sört.

```
INSERT INTO Szeret
VALUES('Zsu', 'Bud');
```

- A reláció neve után megadhatjuk az attribútumait.
- Ennek alapvetően két oka lehet:
  1. elfelejtettük, hogy a reláció definíciójában, milyen sorrendben szerepeltek az attribútumok.
  2. Nincs minden attribútumnak értéke, és azt szeretnénk, ha a hiányzó értékeket NULL vagy default értékkel helyettesítenék.

Példa:

```
INSERT INTO Szeret(sör, név)
VALUES('Bud', 'Zsu');
```

Több sor beszúrása ---1

- Egy lekérdezés eredményét is beszúrhatjuk:

  INSERT INTO <reláció>

  (<alkérdés>);

- A lekérdezést teljesen ki kell értékelni, mielőtt a sorokat beszúrnánk.

- A Látogat(név, söröző) tábla felhasználásával adjuk hozzá a LehetBarát(név) táblához Zsu „lehetséges barátait”, vagyis azokat a sörivókat, akik legalább egy olyan söröző látogatnak, ahova Zsu is szokott járni.

Több sor beszúrása ---2

```
INSERT INTO LehetBarát
(SELECT l2.név
 FROM Látogat l1, Látogat l2
 WHERE l1.név = 'Zsu' AND
       l2.név <> 'Zsu' AND
       l1.söröző = l2.söröző);
```

(SELECT) a másik sörivó

(FROM) névpárok:
az első Zsu,
a második nem Zsu,
de van olyan söröző,
amit mindketten
látogatnak.

Több sor beszúrása ---3

```
INSERT INTO LehetBarát
(SELECT név
 FROM Látogat
 WHERE név <> 'Zsu'
      AND söröző IN
           (SELECT söröző
            FROM Látogat
            WHERE név = 'Zsu' ));
```

(SELECT) a másik sörivó

aki nem Zsu és jár olyan söröző be, ahová Zsu is, előző példa itt alkérdéssel

Törlés (delete)

- A törlendő sorokat egy WHERE feltétel segítségével adjuk meg:

```
DELETE FROM <reláció>
        WHERE <feltétel>;
```

- Példa:

```
DELETE FROM Szeret
        WHERE nev = 'Zsu' AND
          sör = 'Bud';
```

- Az összes sor törlése:

```
DELETE FROM Szeret; --- ~SELECT FROM
```

vagy `DELETE Szeret;` --- ~ UPDATE relációnév

Példa: Több sor törlése

- A Sörök(név, gyártó) táblából töröljük azokat a söröket, amelyekhez létezik olyan sör, amit ugyanaz a cég gyártott.

Példa: Több sor törlése

- A Sörök(név, gyártó) táblából töröljük azokat a söröket, amelyekhez létezik olyan sör, amit ugyanaz a cég gyártott.

```
DELETE FROM Sörök s
WHERE EXISTS (
   SELECT név FROM Sörök
   WHERE gyártó = s.gyártó
        AND név <> s.név);
```

(WHERE) azok a sörök, amelyeknek ugyanaz a gyártója, mint az s éppen aktuális sorának, a nevük viszont különböző.

A törlés szemantikája

- Tegyük fel, hogy az Anheuser-Busch csak Bud és Bud Lite söröket gyárt.
- Tegyük fel még, hogy *s* sorai közt a Bud fordul elő először.
- Az alkérdés nem üres, a későbbi Bud Lite sor miatt, így a Bud törlődik.
- Kérdés: a Bud Lite sor törlődik-e?

A törlés szemantikája

- Válasz: igen, a Bud Lite sora is törlődik.
- A törlés ugyanis két lépésben hajtódik végre.
  1. Kijelöljük azokat a sorokat, amelyekre a WHERE feltétele teljesül.
  2. Majd töröljük a kijelölt sorokat.

Módosítás (update)

- Bizonyos sorok bizonyos attribútumainak módosítása.

  UPDATE <reláció>

  SET <attribútum értékadások listája>

  WHERE <sorokra vonatkozó feltétel>;

- Fecó telefonszámát 555-1212-re változtatjuk (Fecó itt egy sörivó neve):

  UPDATE Sörivók

  SET telefon = '555-1212'

  WHERE név = 'Fecó';

Példa: Több sor módosítása

- Legfeljebb 4 dollárba kerülhessenek a sörök:

```
UPDATE Felszolgál
SET ár = 4.00
WHERE ár > 4.00;
```

- UPDATE esetén is használhatóak alkérdések a WHERE záradékban, és a SET-ben az érték helyén lehet skalár értéket adó alkérdés is!

Tranzakciók az SQL-ben (Tk.6.6.)
Miért van szükség tranzakciókra?

- Az adatbázis rendszereket általában több felhasználó és folyamat használja egyidőben.
  - Lekérdezések és módosítások egyaránt történhetnek.
- Az operációs rendszerektől eltérően, amelyek támogatják folyamatok interakcióját, az adatbázis rendszereknek el kell különíteniük a folyamatokat.

Példa: rossz interakció

- Egy időben ketten töltenek fel 100 dollárt ugyanarra a számlára ATM-en keresztül.
  - Az adatbázis rendszernek biztosítania kell, hogy egyik művelet se vesszen el.
- Ezzel szemben az operációs rendszerek megengedik, hogy egy dokumentumot ketten szerkesszenek egyidőben. Ha mind a ketten írnak, akkor az egyik változtatás elvész (elveszhet).

Tranzakciók

- Tranzakció = olyan folyamat, ami adatbázis lekérdezéseket, módosításokat tartalmaz.
- Az utasítások egy „értelmes egészt” alkotnak.
- Egyetlen utasítást tartalmaznak, vagy az SQL-ben explicit módon megadhatóak.
- Adatbázisok-1 vizsgára csak a tranzakciók 4 ACID tulajdonságát, COMMIT és ROLLBACK utasításokat kell tudni (lásd a köv. két dián), a többi dia nem-prog.inf-es csak egy-féléves kurzusra szól. A prog.inf-en ezt a köv.félévben Adatbázis-2 tanulják majd még részletesebben.

ACID tranzakciók

ACID tulajdonságok:

- Atomiság (atomicity): a tranzakció egységesen lefut vagy nem, vagy az összes vagy egy utasítás sem hajtódik végre.
- Konzisztencia (consistency): a tranzakció futása után konzisztens legyen az adatbázis, megszorításokkal, triggerekkel biztosítjuk.
- Elkülönítés (isolation): párhuzamos végrehajtás eredménye egymás utáni végrehajtással egyezzen meg
- Tartósság (durability): a befejezett tranzakció eredménye rendszerhiba esetén sem veszhet el
- Opcionálisan: gyengébb feltételek is megadhatóak.

COMMIT és ROLLBACK

- A COMMIT utasítás a tranzakció sikeres befejeződését eredményezi. Egy sikeresen befejeződött tranzakció a kezdete óta végrehajtott utasításainak módosításait tartósan rögzíti az adatbázisban
  - vagyis a módosítások **véglegesítődnek**.
- A ROLLBACK utasítás megszakítja a tranzakció végrehajtását, és annak sikertelen befejeződését eredményezi. Az így befejezett tranzakció SQL utasításai által végrehajtott módosításokat a rendszer meg nem történtekké teszi
  - Vagyis az összes utasítás **visszagörgetésre kerül**, a módosítások nem jelennek meg az adatbázisban.

[Példa: egymásra ható folyamatok]

- A Felszolgál(bár, sör, ár) táblánál tegyük fel, hogy Joe bárjában csak Bud és Miller sörök kaphatók 2.50 és 3.00 dollárért.
- Sally a Felszolgál táblából Joe legolcsóbb és legdrágább sörét kérdezi le.
- Joe viszont úgy dönt, hogy a Bud és Miller sörök helyett ezentúl Heinekent árul 3.50 dollárért.

[Sally utasításai]

(max)SELECT MAX(ár) FROM Felszolgál
WHERE bár = 'Joe bárja';

(min) SELECT MIN(ár) FROM Felszolgál
WHERE bár = 'Joe bárja';

[Joe utasításai]

- Ugyanabban a pillanatban Joe a következő utasításokat adja ki:

(del) DELETE FROM Felszolgál
WHERE bár = 'Joe bárja';

(ins) INSERT INTO Felszolgál
VALUES('Joe bárja', 'Heineken', 3.50);

[Átfedésben álló utasítások]

- A (max) utasításnak a (min) kell végrehajtódnia, hasonlóan (del) utasításnak az (ins) előtt, ettől eltekintve viszont nincsenek megszorítások a sorrendre vonatkozóan, ha Sally és Joe utasításait nem gyűjtjük egy-egy tranzakcióba.

[Példa: egy furcsa átfedés]

- Tételezzük fel a következő végrehajtási sorrendet: (max)(del)(ins)(min).

Joe árai:	{2.50,3.00}	{2.50,3.00}		{3.50}
Utasítás:	(max)	(del)	(ins)	(min)
Eredmény:	3.00			3.50

- Mit lát Sally? MAX < MIN!

[Probléma megoldása tranzakciókkal]

- Ha Sally utasításait, (max)(min), egy tranzakcióba gyűjtjük, akkor az előbbi inkonzisztencia nem történhet meg.
- Joe árait ekkor egy adott időpontban látja.
  - Vagy a változtatások előtt vagy utánuk, vagy közben, de a MAX és a MIN ugyanazokból az árakból számolódik.

[Másik hibaforrás: visszagörgetés]

- Tegyük fel, hogy Joe a (del)(ins) és utasításokat nem, mint tranzakció hajtja végre, utána viszont úgy dönt, jobb ha visszagörgeti a módosításokat.
- Ha Sally az (ins) után, de visszagörgetés előtt hajtatja végre a tranzakciót, olyan értéket kap, 3.50, ami nincs is benne az adatbázisban végül.

[Megoldás]

- A (del)(ins) és utasításokat Joe-nak is, mint tranzakciót kell végrehajtatnia, így a változtatások akkor válnak láthatóvá, ha tranzakció egy COMMIT utasítást hajt végre.
  - Ha a tranzakció ehelyett visszagörgetődik, akkor a hatásai sohasem válnak láthatóvá.

[Elkülönítési szintek]

- Az SQL négy elkülönítési szintet definiál, amelyek megmondják, hogy milyen interakciók engedélyezettek az egy időben végrehajtódó tranzakciók közt.
- Ezek közül egy szint (“sorbarendezhető”) = ACID tranzakciók.
- Minden ab rendszer a saját tetszése szerint implementálhatja a tranzakciókat.

[Elkülönítési szint megválasztása]

- Az utasítás:

SET TRANSACTION ISOLATION LEVEL *X*

ahol *X* =

1. SERIALIZABLE
2. REPEATABLE READ
3. READ COMMITTED
4. READ UNCOMMITTED

[Sorbarendezhető tranzakciók]

- Ha Sally a (max)(min), Joe a (del)(ins) tranzakciót hajtatja végre, és Sally tranzakciója SERIALIZABLE elkülönítési szinten fut, akkor az adatbázist vagy Joe módosításai előtt vagy után látja, a (del) és (ins) közötti állapotban sohasem.

[Az elkülönítési szint választása]

- Ez a döntés csak azt mondja meg, hogy az illető hogyan látja az adatbázist, és nem azt, hogy mások hogy látják azt.
- Példa: Ha Joe sorbarendezhető elkülönítési szintet használ, de Sally nem, akkor lehet, hogy Sally nem talál árakat Joe bárja mellett.
  - azaz, mintha Sally Joe tranzakciójának közepén futtatná a sajátját.

[Read-Commited tranzakciók]

- Ha Sally READ COMMITTED elkülönítési szintet választ, akkor csak kommitálás utáni adatot láthat, de nem feltétlenül mindig ugyanazt az adatot.
- Példa: READ COMMITTED mellett megengedett a (max)(del)(ins)(min) átfedés amennyiben Joe kommitál.
  - Sally legnagyobb megdöbbenésére: MAX < MIN.

[Repeatable-Read tranzakciók]

- Hasonló a read-commited megszorításhoz. Itt, ha az adatot újra beolvassuk, akkor amit először láttunk, másodszor is látni fogjuk.
- De második és az azt követő beolvasások után akár *több* sort is láthatunk.

[Példa: ismételhető olvasás]

- Tegyük fel, hogy Sally REPEATABLE READ elkülönítési szintet választ, a végrehajtás sorrendje: (max)(del)(ins)(min).
  - (max) a 2.50 és 3.00 dollár árakat látja.
  - (min) látja a 3.50 dollárt, de 2.50 és 3.00 árakat is látja, mert egy korábbi olvasáskor (max) már látta azokat.

SQL DDL
Adatbázis relációsémák definiálása

- Az SQL tartalmaz adatleíró részt (DDL), az adatbázis objektumainak a leírására és megváltoztatására. Objektumok leíró parancsa a CREATE utasítás.
- A relációt az SQL-ben táblának (TABLE) nevezik, az SQL alapvetően háromféle táblát kezel:
  - Alaptáblák (permanens) CREATE TABLE
  - Nézettáblák CREATE VIEW
  - Átmeneti munkatáblák (WITH utasítás)
- Alaptáblák megadása: CREATE TABLE

Tábla/reláció sémák SQL-ben

- A legegyszerűbb formája:

  CREATE TABLE relációnév (
  Attribútum deklarációk listája,
  Kiegészítő lehetőségek
  );

- Az attribútum deklaráció legalapvetőbb elemei:

  Attribútumnév típus [kiegészítő lehetőségek]

-- itt: a **típus** olyan, amit az SQL konkrét megvalósítása támogat (gyakorlaton Oracle környezetben nézzük meg), Típusok, pl: INTEGER, REAL, CHAR, VARCHAR, DATE

-- **Kiegészítő lehetőségek** például [DEFAULT], [UNIQUE], [PRIMARY KEY], [FOREIGN KEY, REFERENCES], stb.

Egyszerű példák táblák létrehozására

```
CREATE TABLE Sörözők (
    név CHAR(20),
    város VARCHAR2(40),
    tulaj CHAR(30),
    engedély DATE  DEFAULT SYSDATE
    );

CREATE TABLE Felszolgál (
          söröző CHAR(20),
          sör   VARCHAR2(20),
          ár   NUMBER(10,2) DEFAULT 100
    );
```

Az SQL értékekről (bővebben gyakorlaton)

- INTEGER, REAL, stb, a szokásos értékek, számok.
- STRING szintén, de itt egyes-aposztróf közé kell tenni a ‘szöveget’ (vagyis nem „macskaköröm” közé).

  Két egyes-aposztróf = egynek felel meg, például `'Joe''s Bar'` megfelel a Joe’s Bar szövegnek.
- DATE és TIME típusok is vannak az SQL-ben.
- A dátum formátumát meg kell adni DATE ’yyyy-mm-dd’

  Például: `DATE '2007-09-30'` (2007. szept. 30)
- Az idő formátumát is meg kell adni TIME ’hh:mm:ss’

  Például: `TIME '15:30:02.5'` (délután fél 4 múlt két és fél másodperccel)
- Bármely érték lehet NULL hiányzó érték:

Hiányzó értékek: NULL

- Az SQL lehetővé teszi a táblákban a hiányzó értékeket, vagyis a relációk soraiban az attribútum értéke ne legyen megadva, hanem egy speciális NULL nullérték legyen.
- **A nullérték értelmezésére** több lehetőségünk is van:
  - Nem-ismert érték: például tudom, „Joe's Bár"-jának van valamilyen címe, de nem tudom, hogy mi az.
  - Nem-definiált érték: például a házastárs attribútumnak egyedülálló embereknél nincs olyan értéke, aminek itt értelme lenne, nincs házastársa, ezért nullérték.
  - stb (van olyan cikk, amely több százféle okot felsorol)

Default értékek megadása

- A CREATE TABLE utasításban az oszlopnevet DEFAULT kulcsszó követheti és egy érték.
- Ha egy beszúrt sorban hiányzik az adott attribútum értéke, akkor a default értéket kapja.

```
CREATE TABLE Sörivók(
    név CHAR(30)
    cím CHAR(50) DEFAULT 'Sesame St'
    telefon CHAR(16)  );
INSERT INTO Sörivók(név)
    VALUES('Zsu');
```

Az eredmény sor:

név	cím	telefon
Zsu	Sesame St	NULL

Megszorítások (áttekintés)

(1) Kulcsok és idegen kulcsok
- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

(2) Értékekre vonatkozó feltételek
- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

(3) Sorokra vonatkozó megszorítások
- Sorra vonatkozó CHECK feltételek

(4) Megszorítások módosítása (constraints)

(5) Önálló megszorítások (create assertion)

(6) Triggerek (create trigger)

Tankönyv példa: Filmek séma

Filmek(
cím:string,
év:integer,
hossz:integer,
műfaj:string,
stúdióNév:string,
producerAzon:integer)

FilmSzínész(
név:string,
cím:string,
nem:char,
születésiDátum:date)

Stúdió(
név:string,
cím:string,
elnökAzon:integer)

SzerepelBenne(
filmCím:string,
filmÉv:integer,
színészNév:string)

GyártásIrányító(
név:string,
cím:string,
azonosító:integer,
nettóBevétel:integer)

Mit jelentenek az aláhúzások?

Tankönyv példája, hibás fordítás: title=(film)cím és address=(lak)cím
Tervezéssel később foglalkozunk, ez a példa hibás, az elnevezések, de így jó lesz, hogy a lekérdezéseknél megnézzük hogyan kezeljük.

Példa megszorításokra: Kulcs

- Előző példában: attribútumok aláhúzása mit jelent?
- Filmek: elvárjuk, hogy ne legyen a megengedett előfordulásokban két különböző sor, amelyek megegyeznek cím, év attribútumokon.
- Egyszerű kulcs egy attribútumból áll, de egy kulcs nem feltétlenül áll egy attribútumból, ez az összetett kulcs. Például a **Filmek** táblában a cím és év együtt alkotják a kulcsot, nem elég a cím, ugyanis van például (King Kong, 1933), (King Kong, 1976) és (King Kong, 2005).
- A kulcsot aláhúzás jelöli: Filmek (cím, év, hossz, …)

Kulcs megadása

- PRIMARY KEY vagy UNIQUE
- Nincs a relációnak két olyan sora, amely a lista minden attribútumán megegyezne.
- Kulcs esetén nincs értelme a DEFAULT értéknek.
- Kulcsok megadásának két változata van:
  - Egyszerű kulcs (egy attribútum) vagy
  - Összetett kulcs (attribútumok listája)

Egyszerű kulcs megadása

- Ha a kulcs egyetlen attribútum, akkor ez az attribútum deklarációban megadható

  <attribútumnév> <típus> PRIMARY KEY

  vagy <attribútumnév> <típus> UNIQUE

- Példa:

```
CREATE TABLE Sörök (
        név         CHAR(20) UNIQUE,
        gyártó      CHAR(20)
);
```

Összetett kulcs megadása

- Ha a kulcs több attribútumból áll, akkor a CREATE TABLE utasításban az attribútum deklaráció után a kiegészítő részben meg lehet adni további tábla elemeket: PRIMARY KEY (attrnév$_1$, … attrnév$_k$)
- Példa:

```
CREATE TABLE Felszolgál (
        söröző      CHAR(20),
        sör         VARCHAR2(20),
        ár          NUMBER(10,2),
        PRIMARY KEY (söröző, sör)
);
```

PRIMARY KEY vs. UNIQUE

- Csak egyetlen PRIMARY KEY lehet a relációban, viszont UNIQUE több is lehet.
- PRIMARY KEY egyik attribútuma sem lehet NULL egyik sorban sem. Viszont UNIQUE-nak deklarált attribútum lehet NULL, vagyis a táblának lehet olyan sora, ahol a UNIQUE attribútum értéke NULL vagyis hiányzó érték.
- az SQL lekérdezésnél adjuk meg hogyan kell ezzel a speciális értékkel gazdálkodni, hogyan lehet NULL-t kifejezésekben és hogyan lehet feltételekben használni
- Következő héten visszatérünk a megszorítások és a hivatkozási épség megadására.

Idegen kulcsok megadása

- Az első előadáson a táblák létrehozásához veszünk kiegészítő lehetőségeket: Kulcs és idegen kulcs (foreign key) hivatkozási épség megadása
- Az egyik tábla egyik oszlopában szereplő értékeknek szerepelnie kell egy másik tábla bizonyos attribútumának az értékei között.
- A hivatkozott attribútumoknak a másik táblában kulcsnak kell lennie! (PRIMARY KEY vagy UNIQUE)
- Példa: **Felszolgál(söröző, sör, ár)** táblára megszorítás, hogy a sör oszlopában szereplő értékek szerepeljenek a **Sörök(név, gyártó)** táblában a név oszlop értékei között.

Idegen kulcs megadása: attribútumként

REFERENCES kulcsszó használatának két lehetősége: attribútumként vagy sémaelemként lehet megadni.

1.) Attribútumonként (egy attribútumból álló kulcsra)

Példa:

```
CREATE TABLE Sörök (
    név     CHAR(20) PRIMARY KEY,
    gyártó CHAR(20) );

CREATE TABLE Felszolgál (
    söröző CHAR(20),
    sör   CHAR(20) REFERENCES Sörök(név),
    ár    REAL );
```

Idegen kulcs megadása: sémaelemként

2.) Sémaelemként (egy vagy több attr.-ból álló kulcsra)

FOREIGN KEY (attribútum lista)

REFERENCES relációnév (attribútum lista)

Példa:

```
CREATE TABLE Sörök (
  név       CHAR(20),
  gyártó    CHAR(20),
  PRIMARY KEY (név) );

CREATE TABLE Felszolgál (
  söröző    CHAR(20),
  sör       CHAR(20),
  ár        REAL,
  FOREIGN KEY(sör) REFERENCES Sörök(név));
```

Hivatkozási épség, idegen kulcs megszorítások megőrzése

- Példa: *R* = Felszolgál, *S* = Sörök.
- Egy idegen kulcs megszorítás *R* relációról *S* relációra kétféleképpen sérülhet:
  1. Egy *R*-be történő beszúrásnál vagy *R*-ben történő módosításnál *S*-ben nem szereplő értéket adunk meg.
  2. Egy *S*-beli törlés vagy módosítás „lógó” sorokat eredményez *R*-ben.

Hogyan védekezzünk? --- (1)

- Példa: *R* = Felszolgál, *S* = Sörök.
- Nem engedjük, hogy Felszolgál táblába a Sörök táblában nem szereplő sört szúrjanak be vagy Sörök táblában nem szereplő sörre módosítsák (nincs választási lehetőségünk, a rendszer visszautasítja a megszorítást sértő utasítást)
- A Sörök táblából való törlés vagy módosítás, ami a Felszolgál tábla sorait is érintheti (mert sérül az idegen kulcs megszorítás) 3-féle módon kezelhető (lásd köv.oldal)

Hogyan védekezzünk? --- (2)

1. Alapértelmezés (Default) : a rendszer nem hajtja végre a törlést.
2. Továbbgyűrűzés (Cascade): a Felszolgál tábla értékeit igazítjuk a változáshoz.
   - Sör törlése: töröljük a Felszolgál tábla megfelelő sorait.
   - Sör módosítása: a Felszolgál táblában is változik az érték.
3. Set NULL: a sör értékét állítsuk NULL-ra az érintett sorokban.

Példa: továbbgyűrűzés

- Töröljük a Bud sort a Sörök táblából:
  - az összes sort töröljük a Felszolgál táblából, ahol sör oszlop értéke ’Bud’.
- A ’Bud’ nevet ’Budweiser’-re változtatjuk:
  - a Felszolgál tábla soraiban is végrehajtjuk ugyanezt a változtatást.

Példa: Set NULL

- A Bud sort töröljük a Sörök táblából:
  - a Felszolgál tábla sör = 'Bud' soraiban a Budot cseréljük NULL-ra.
- 'Bud'-ról 'Budweiser'-re módosítunk:
  - ugyanazt kell tennünk, mint törléskor.

A stratégia kiválasztása

- Ha egy idegen kulcsot deklarálunk megadhatjuk a SET NULL és a CASCADE stratégiát is beszúrásra és törlésre is egyaránt.
- Az idegen kulcs deklarálása után ezt kell írnunk:

  ON [UPDATE, DELETE][SET NULL CASCADE]
- Ha ezt nem adjuk meg, a default stratégia működik.

Példa: stratégia beállítása

```
CREATE TABLE Felszolgál (
  söröző CHAR(20),
  sör          CHAR(20),
  ár           REAL,
  FOREIGN KEY(sör)
     REFERENCES Sörök(név)
     ON DELETE SET NULL
     ON UPDATE CASCADE
);
```

Megszorítások ellenőrzésének késleltetése

- Körkörös megszorítások miatt szükség lehet arra, hogy a megszorításokat ne ellenőrizze, amíg az egész tranzakció be nem fejeződött.
- Bármelyik megszorítás deklarálható DEFERRABLE (késleltethető) vagy NOT DEFERRABLE-ként (vagyis minden adatbázis módosításkor a megszorítás közvetlenül utána ellenőrzésre kerül). DEFERRABLE-ként deklaráljuk, akkor lehetőségünk van arra, hogy a megszorítás ellenőrzésével várjon a rendszer a tranzakció végéig.
- Ha egy megszorítás késleltethető, akkor lehet
  - INITIALLY DEFERRED (az ellenőrzés a tranzakció jóváhagyásáig késleltetve lesz) vagy
  - INITIALLY IMMEDIATE (minden utasítás után ellenőrzi)

Értékekre vonatkozó feltételek

- Egy adott oszlop értékeire vonatkozóan adhatunk meg megszorításokat.
- A CREATE TABLE utasításban az attribútum deklarációban NOT NULL kulcsszóval
- az attribútum deklarációban CHECK(<feltétel>) A feltétel, mint a WHERE feltétel, alkérdés is használható. A feltételben csak az adott attribútum neve szerepelhet, más attribútumok (más relációk attribútumai is) csak alkérdésben szerepelhetnek.

Példa: értékekre vonatkozó feltétel

```
CREATE TABLE Felszolgál (
  söröző CHAR(20) NOT NULL,
  sör          CHAR(20) CHECK ( sör IN
                (SELECT név FROM Sörök)),
  ár            REAL CHECK ( ár <= 5.00 )
  );
```

Mikor ellenőrzi?

- Érték-alapú ellenőrzést csak beszúrásnál és módosításnál hajt végre a rendszer.
  - **Példa:** `CHECK (ár <= 5.00)` a beszúrt vagy módosított sor értéke nagyobb 5, a rendszer nem hajtja végre az utasítást.
  - **Példa:** `CHECK (sör IN (SELECT név FROM Sörök)`, ha a Sörök táblából törlünk, ezt a feltételt nem ellenőrzi a rendszer.

Sorokra vonatkozó megszorítások

- A CHECK (<feltétel>) megszorítás a séma elemeként is megadható.
- A feltételben tetszőleges oszlop és reláció szerepelhet.
  - De más relációk attribútumai csak alkérdésben jelenhetnek meg.
- Csak beszúrásnál és módosításnál ellenőrzi a rendszer.

Példa: sor-alapú megszorítások

- Csak Joe bárja nevű söröző̋ben lehetnek drágábbak a sörök 5 dollárnál:

```
CREATE TABLE Felszolgál (
    söröző  CHAR(20),
    sör     CHAR(20),
    ár      REAL,
    CHECK (söröző= 'Joe bárja'
                OR ár <= 5.00)
);
```

Megszorítások elnevezése

- Nevet tudunk adni a megszorításoknak, amire később tudunk hivatkozni (könnyebben lehet később majd törölni, módosítani)
- név CHAR(30) CONSTRAINT NévKulcs
  PRIMARY KEY,
- nem CHAR(1) CONSTRAINT FérfiVagyNő
  CHECK (nem IN ('F', 'N')),
- CONSTRAINT Titulus
  CHECK (nem = 'N' OR név NOT LIKE 'Ms.\%')

Megszorítások módosítása

- ALTER TABLE FilmSzínész ADD CONSTRAINT NévKulcs PRIMARY KEY (név);
- ALTER TABLE FilmSzínész ADD CONSTRAINT FérfiVagyNő CHECK (nem IN ('F', 'N'));
- ALTER TABLE FilmSzínész ADD CONSTRAINT Titulus CHECK (nem = 'N' OR név NOT LIKE 'Ms.\%');

Összefoglalás: Kulcsok, idegen kulcsok

- Kiegészítő lehetőségeket: Kulcs és idegen kulcs (foreign key) hivatkozási épség megadása
- PRIMARY KEY vagy UNIQUE
- Kulcsok megadásának két változata van:
  - Egyszerű kulcs (egy attribútum) vagy
  - Összetett kulcs (attribútumok listája)
- FOREIGN KEY, a hivatkozott attribútumoknak a másik táblában kulcsnak kell lennie!
- REFERENCES kulcsszó használatának két lehetősége: attribútumként vagy sémaelemként lehet megadni.

Összefoglalás: A stratégia kiválasztása hivatkozási épség megőrzésére

- Ha idegen kulcsot deklarálunk megadhatjuk beszúrásra is és törlésre is a SET NULL és a CASCADE stratégiát:

  ON [UPDATE, DELETE][SET NULL CASCADE]

- Ha nem adjuk meg, a default stratégia működik, elutasítja az épséget sértő törlést vagy módosítást.
- Példa:

```
CREATE TABLE Felszolgál (
söröző CHAR(20), sör CHAR(20), ár REAL,
FOREIGN KEY(sör)REFERENCES Sörök(név)
    ON DELETE SET NULL
    ON UPDATE CASCADE);
```

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?

Feladatok

- Házi feladat: Gyakorlás az Oracle Példatár feladatai:
- DML-utasítások, tranzakciók
  - DML-utasítások: insert, update, delete (Példatár 5.fej.)
  - Adatbázis-tranzakciók: commit, rollback, savepoint
- DDL-utasítások, create table
  - DDL-utasítások: adattáblák létrehozása, módosítása, integritási megszorítások (Példatár 5.fejezet folyt.) és
  - Nézettábla létrehozása és törlése, táblák tartalmának módosítása nézettáblákon keresztül (Példatár 6.fej.)